DEVELOPING CUSTOM SQL QUERIES

Service Builder's finder methods facilitate searching for entities by their attributes–their column values. Add the column as a parameter for the finder in your `service.xml` file, run Service Builder, and it generates the finder method in your persistence layer and adds methods to your service layer that invoke the finder. But what if you'd like to do more complicated searches that incorporate attributes from multiple entities?

For example, consider the Event Listing (https://github.com/liferay/liferay-docs/tree/6.2.x/develop/tutorials/code/tutorials-sdk/portlets/event-listing-portlet) portlet. Suppose you want to find an event based on its name, description, and location name. In the Event Listing portlet, the event entity refers to its location by the location's ID, not its name. That is, the event entity table, `Event_Event`, refers to an event's location by its long integer ID in the table's `locationId` column. But you need to access the *name* of the event's location. Of course, with SQL you can join the event and location tables to include the location name. But how would you incorporate custom SQL into your portlet? And how would you invoke the SQL from your service? Service Builder lets you do this by specifying the SQL as *Liferay custom SQL* and invoking it in your service via a *custom finder method*.

Liferay custom SQL is a Service Builder-supported method for performing complex and custom queries against the database. Invoking custom SQL from a finder method in your persistence layer is straightforward. And Service Builder helps you generate the interfaces to your finder method. It's easy to do by following these steps:

1. Specify your custom SQL.
2. Implement your finder method.
3. Access your finder method from your service.

Next, using the Event Listing portlet as an example, you'll learn how to accomplish these steps.

STEP 1: SPECIFY YOUR CUSTOM

/-/knowledge_base/6-2/plugins-sdk)

Developing Plugins with Liferay IDE (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-ide)

Developing with Maven (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/maven)

Deploying Plugins (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/deploying-plugins)

MVC Portlets (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/developing-jsp-portlets-using-liferay-mvc)

JSF Portlets with Liferay Faces (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/jsf-portlets-with-liferay-faces)

Service Builder and Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/service-builder)

What is Service Builder? (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/what-is-service-builder)

SQL

After you've tested your SQL, you must specify it in a particular file for Liferay to access it. Liferay's `CustomSQLUtil` class looks up custom SQL from a file called `default.xml` in your portlet project's `docroot/WEB-INF/src/custom-sql/` folder. You must create the `custom-sql` folder and create the `default.xml` file in that `custom-sql` folder. The `default.xml` file must adhere to the following format:

```
<custom-sql>
    <sql id="[fully-qualified class name + method]">
    SQL query wrapped in <![CDATA[...]]>
    No terminating semi-colon
    </sql>
</custom-sql>
```

You can add a `custom-sql` element for every custom SQL query you'd like for your portlet, as long as each query has a unique ID. The convention we recommend using for the ID value is the fully-qualified class name of the finder followed by a dot (`.`) character and the name of the finder method. More detail on the finder class and finder methods is in Step 2.

In the Event Listing project, the following ID value was used for the query:

```
com.liferay.docs.eventlisting.service.persistence.\
EventFinder.findByEventNameEventDescriptionLocationName
```

Custom SQL must be wrapped in character data (`CDATA`) for the `sql` element. Importantly, the SQL must *not* be terminated with a semi-colon. Following these rules, the `default.xml` file of the Event Listing project specifies an SQL query that joins the Event and Location tables:

```
<?xml version="1.0" encoding="UTF-8"?>
<custom-sql>
    <sql id="com.liferay.docs.eventlisting.service.persistence.EventFi
nder.\
findByEventNameEventDescriptionLocationName">
<![CDATA[
        SELECT Event_Event.*
        FROM Event_Event
        INNER JOIN
            Event_Location ON Event_Event.locationId = Event_Location.
locationId
        WHERE
            (Event_Event.name LIKE ?) AND
            (Event_Event.description LIKE ?) AND
            (Event_Location.name LIKE ?)
]]>
    </sql>
</custom-sql>
```

Defining an Object-Relational Map with Service Builder (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/defining-an-object-relational-map-with-service-builder)

Running Service Builder and Understanding the Generated Code (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/running-service-builder-and-understanding-the-generated-code)

Understanding ServiceContext (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/servicecontext)

Creating Local Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/writing-local-service-classes)

Invoking Local Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/invoking-local-services)

Creating Remote Services (https://dev.liferay.com/develop/tutorials

If you copy the XML fragment above, make sure to delete the backslash (`\`) character from the end of the ID so that the finder method name `findByEventNameEventDescriptionLocationName` immediately follows the package path specified below:

```
com.liferay.docs.eventlisting.service.persistence.
```

Now that you've specified some custom SQL, the next step is to implement a finder method to invoke it. The method name for the finder should match the ID you just specified for the `sql` element.

STEP 2: IMPLEMENT YOUR FINDER METHOD

After specifying your custom SQL query, you need to implement the finder method to invoke it. This should be done in the service's persistence layer. Service Builder generates the interface for the finder but you need to create the implementation.

The first step is to create a `*FinderImpl` class in the service persistence package. The Event Listing project includes the `EventFinderImpl` class in the `com.liferay.docs.eventlisting.service.persistence.impl` package. Your class, like `EventFinderImpl`, should extend `BasePersistenceImpl<Event>`.

Run Service Builder to generate the `*Finder` interface and the `*Util` class for the finder. Service Builder generates the `*Finder` interface and the `*FinderUtil` utility class based on the `*FinderImpl` class. Modify your `*FinderImpl` class to have it implement the `*Finder` interface you just generated:

```
public class *FinderImpl extends BasePersistenceImpl<Event>
    implements EventFinder {


}
```

Now you can create our finder method in your `EventFinderImpl` class. Add your finder method and static field to the `*FinderImpl` class. Here's the `EventFinderImpl` class:

/-/knowledge_base/6-2/creating-remote-services)

Invoking Remote Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/invoking-remote-services)

Service Security Layers (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/service-security-layers)

Finding and Invoking Liferay Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/finding-and-invoking-liferay-services)

Registering JSON Web Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/registering-json-web-services)

Invoking JSON Web Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/invoking-json-web-services)

JSON Web Services Invoker (https://dev.liferay.com/develop/tutorials/-/knowledge_base

```
public List<Event> findByEventNameEventDescriptionLocationName(
    String eventName, String eventDescription, String locationName,
    int begin, int end) {

    Session session = null;
    try {
        session = openSession();

        String sql = CustomSQLUtil.get(
            FIND_BY_EVENTNAME_EVENTDESCRIPTON_LOCATIONNAME);

        SQLQuery q = session.createSQLQuery(sql);
        q.setCacheable(false);
        q.addEntity("Event_Event", EventImpl.class);

        QueryPos qPos = QueryPos.getInstance(q);
        qPos.add(eventName);
        qPos.add(eventDescription);
        qPos.add(locationName);

        return (List<Event>) QueryUtil.list(q, getDialect(), begin, en
d);
    } catch (Exception e) {
        try {
            throw new SystemException(e);
        } catch (SystemException se) {
            se.printStackTrace();
        }
    } finally {
        closeSession(session);
    }

    return null;
}

public static final String FIND_BY_EVENTNAME_EVENTDESCRIPTON_LOCATIONN
AME =
    EventFinder.class.getName() +
        ".findByEventNameEventDescriptionLocationName";
```

Remember to import the required classes. The following imports are required for `EventFinderImpl`:

/6-2/json-web-services-invoker)

JSON Web Services Invocation Examples (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/json-web-services-invocation-examples)

Portal Configuration of JSON Web Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/portal-configuration-of-json-web-services)

Invoking Services Using Skinny JSON Provider (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/invoking-services-using-skinny-json-provider)

SOAP Web Services (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/soap-web-services)

Authorizing Access to Services with OAuth (https://dev.liferay.com/develop/tutorials

```
import java.util.List;

import com.liferay.docs.eventlisting.model.Event;
import com.liferay.docs.eventlisting.model.impl.EventImpl;
import com.liferay.portal.kernel.dao.orm.QueryPos;
import com.liferay.portal.kernel.dao.orm.QueryUtil;
import com.liferay.portal.kernel.dao.orm.SQLQuery;
import com.liferay.portal.kernel.dao.orm.Session;
import com.liferay.portal.kernel.exception.SystemException;
import com.liferay.portal.service.persistence.impl.BasePersistenceImpl
;
import com.liferay.util.dao.orm.CustomSQLUtil;
```

The custom finder method opens a new Hibernate session and uses Liferay's `CustomSQLUtil.get(String id)` method to get the custom SQL to use for the database query. The `FIND_BY_EVENTNAME_EVENTDESCRIPTON_LOCATIONNAME` static field contains the custom SQL query's ID. The `FIND_BY_EVENTNAME_EVENTDESCRIPTON_LOCATIONNAME` string is based on the fully-qualified class name of the `*Finder` interface (`EventFinder`) and the name of the finder method (`findByEventNameEventDescriptionLocationName`).

Awesome! Your custom SQL is in place and your finder method is implemented. Next, you'll call the finder method from your service.

STEP 3: ACCESS YOUR FINDER METHOD FROM YOUR SERVICE

So far, you created a `*FinderImpl` class and generated a `*FinderUtil` utility class. However, your portlet class should not use the finder utility class directly; only a local or remote service implementation (i.e., `*LocalServiceImpl` or `*ServiceImpl`) in your plugin project should invoke the `*FinderUtil` class. This encourages a proper separation of concerns: the portlet classes invoke business logic of the services and the services in turn access the data model using the persistence layer's finder classes. So you'll add a method in the `*LocalServiceImpl` class that invokes the finder method implementation via the `*FinderUtil` class. Then you'll provide the portlet and JSPs access to this service method by rebuilding the service.

The following method in `EventLocalServiceImpl` invokes the finder method discussed in step 2:

```
public List<Event> findByEventNameEventDescriptionLocationName(String
eventName,
    String eventDescription, String locationName, int begin, int end)
throws SystemException {

    return EventFinderUtil.findByEventNameEventDescriptionLocationName
(
        eventName, eventDescription, locationName, begin, end);
}
```

After you've added a service method to invoke your finder method, run Service Builder to generate the interface and make your finder service method available in the `EventLocalServiceUtil` class.

Now you can indirectly call the finder method from your portlet class or from a JSP. To call the finder method in the Event Listing project, just calling `EventLocalServiceUtil.findByEventNameEventDescriptionLocationName(...)`!

Congratulations on developing a custom SQL query and custom finder for your portlet!

RELATED TOPICS

Writing a Data Driven Application (/develop/tutorials/-/knowledge_base/6-2/writing-a-data-driven-application)

Running Service Builder and Understanding the Generated Code (/develop/tutorials/-/knowledge_base/6-2/running-service-builder-and-understanding-the-generated-code)

0 (0 Votes)

Customizing Model Entities Wi... (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/customizing-model-entities-with-model-hints)

Leveraging Hibernate's Criteri... (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/leveraging-hibernates-criteria-api)

Search and Indexing (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/search-and-indexing)

Localization (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/localization)

Asset Framework (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/asset-framework)

Recycle Bin (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/recycle-bin)

Message Bus (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/using-liferays-message-bus)

Workflow (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/workflow)

JavaScript in Liferay (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/javascript-in-liferay)

User Interfaces with AlloyUI (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/alloyui)

User Interfaces with the Liferay UI Taglib

(https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-ui-taglibs)

Liferay Faces Alloy UI Components (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-faces-alloy-ui-components)

Liferay Faces Portal UI Components (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-faces-portal-ui-components)

Liferay Faces Bridge UI Components (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-faces-bridge-ui-components)

Android Apps with Liferay Screens (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/android-apps-with-liferay-screens)

iOS Apps with Liferay Screens (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/ios-apps-with-liferay-screens)

Mobile SDK (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/mobile)

Using the Device Recognition API

(https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/using-the-device-recognition-api)

OpenSocial Gadgets (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/opensocial-gadgets)

Themes and Layout Templates (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/themes-and-layout-templates)

Application Display Templates (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/application-display-templates)

Customizing Liferay Portal with Hooks (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/customizing-liferay-portal)

Audience Targeting (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/audience-targeting)

Importing Resources (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/leveraging-the-resources-importer)

Modularization with OSGi Plugins

(https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/developing-osgi-plugins-for-liferay)

Plugin Security and PACL (https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/plugin-security-and-pacl)

DOWNLOADS

Portal (http://www.liferay.com/downloads/liferay-portal/available-releases)

Social Office (http://www.liferay.com/downloads/social-office/available-releases)

Sync (http://www.liferay.com/downloads/liferay-sync)

Liferay Faces (http://www.liferay.com/community/liferay-projects/liferay-faces/download)

OTHER LIFERAY SITES

(http://www.liferay.com) (http://alloyui.com) (http://issues.liferay.com

PRIVACY POLICY (HTTPS://WWW.LIFERAY.COM/ABOUT-US/PRIVACY)

/

MEET THE TEAM (/MEET-THE-TEAM)